

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии  
Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**  
**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1**  
**дисциплины**  
**«Искусственный интеллект и машинное обучение»**  
**Вариант 12**

Выполнил:  
Рябинин Егор Алексеевич  
2 курс, группа ИВТ-б-о-23-2,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

Проверил:  
Доцент департамента цифровых,  
робототехнических систем и  
электроники института перспективной  
инженерии  
Воронкин Роман Александрович

---

(подпись)

Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_

Ставрополь, 2025 г

**Тема:** Работа с Jupyter Notebook, JupyterLab и Google Colab

**Цель:** исследовать базовые возможности интерактивных оболочек Jupyter Notebook, JupyterLab и Google Colab для языка программирования Python.

**Порядок выполнения работы:**

**Работа в Jupyter Notebook.**

Создадим ноутбук, воспользовавшись кнопкой New, выбрав “Python [conda env:base]\*”.

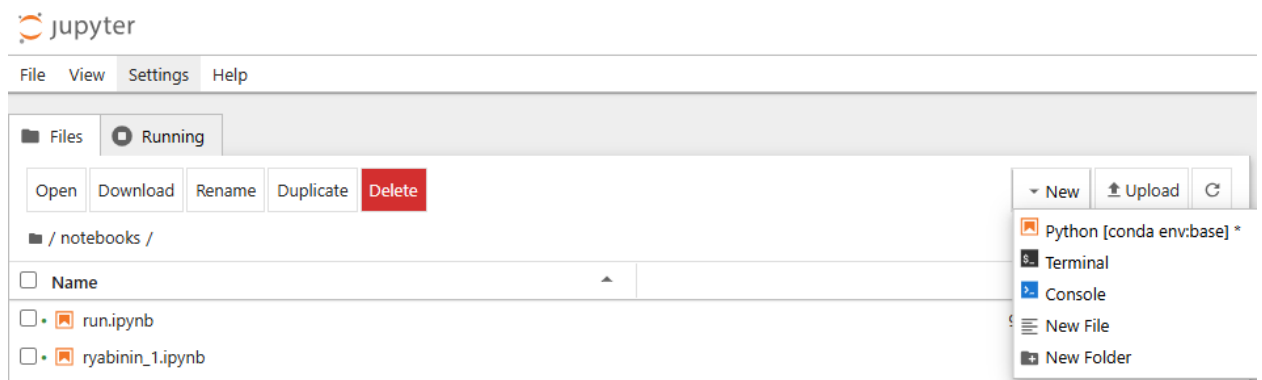


Рисунок 1 – Создание ноутбука

Напишем небольшую программу и выполним ее, существует несколько способов запуска ячеек:

Ctrl+Enter – выполнить содержимое ячейки.

Shift+Enter - выполнить содержимое ячейки и перейти на ячейку ниже.

Alt+Enter – выполнить содержимое ячейки и вставить новую ячейку ниже.

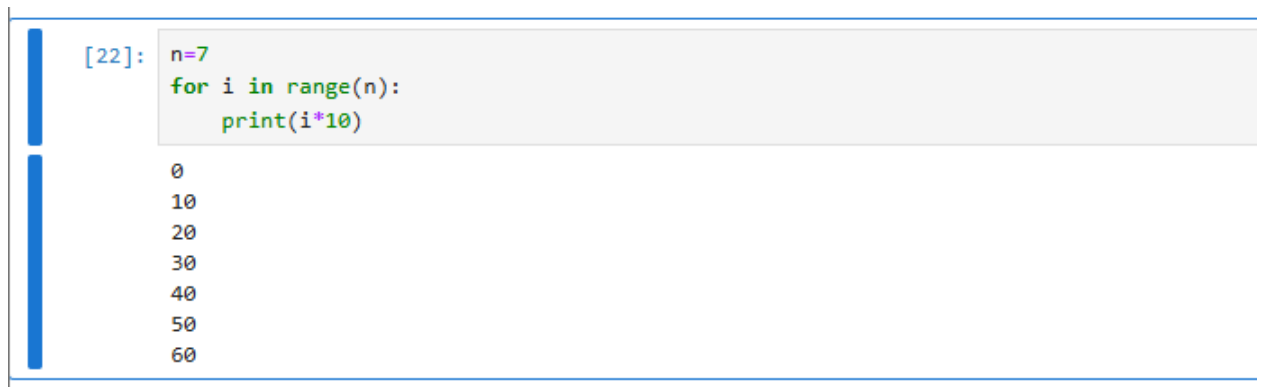


Рисунок 2 – Выполнение ячейки с помощью Ctrl+Enter

```
[44]: n=7
      for i in range(n):
          print(i*10)

0
10
20
30
40
50
60
```

---

```
[42]: print('test')

test
```

---

Рисунок 3 – Выполнение ячейки с помощью Shift+Enter

```
[46]: n=7
      for i in range(n):
          print(i*10)

0
10
20
30
40
50
60
```

---

```
[ ]:
```

---

```
[42]: print('test')

test
```

---

Рисунок 4 – Выполнение ячейки с помощью Alt+Enter

Выведем изображение (график) в ноутбук, для этого выполним команду `%matplotlib inline`, по умолчанию графики Matplotlib открываются в отдельном окне или вкладке веб-браузера, с помощью опции `inline` график отображается непосредственно в ячейке ноутбука.

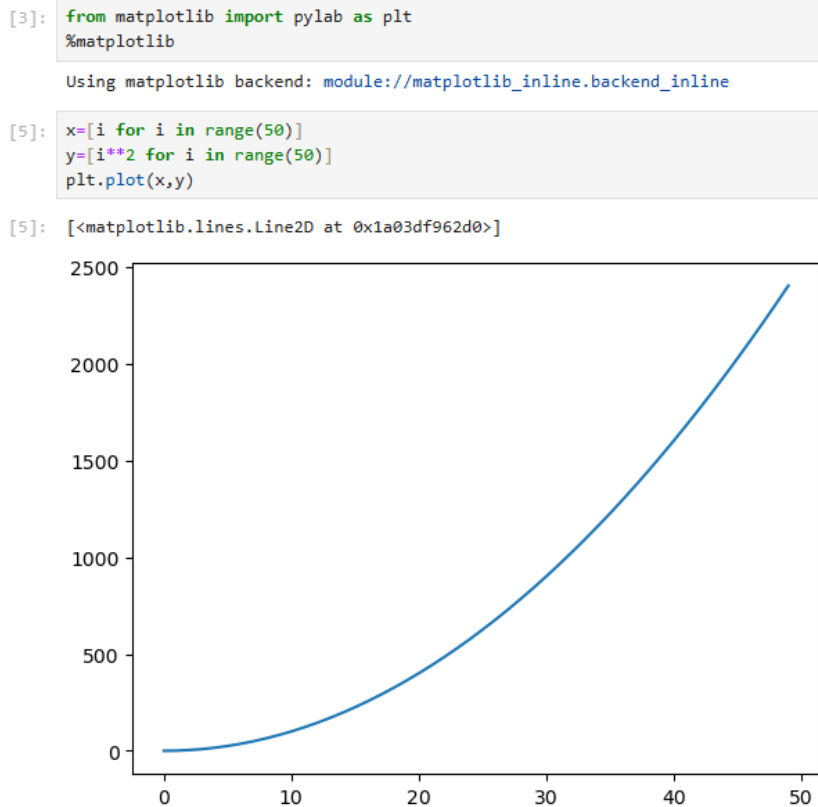


Рисунок 5 – Вывод графика Matplotlib в ячейку ноутбука

Выполним магические команды, т.е. дополнительные команды, выполняемые в рамках оболочки, которые облегчают процесс разработки и расширяют наши возможности.

```
[11]: %lsmagic

[11]: ☒ root
      ☒ line
      automagic "AutoMagics"
      autocall "AutoMagics"
      alias_magic "BasicMagics"
      lsmagic "BasicMagics"
      magic "BasicMagics"
      page "BasicMagics"
      pprint "BasicMagics"
      colors "BasicMagics"
      xmode "BasicMagics"
      quickref "BasicMagics"
      doctest_mode "BasicMagics"
      gui "BasicMagics"
      precision "BasicMagics"
      notebook "BasicMagics"
      save "CodeMagics"
      pastebin "CodeMagics"
      loadnb "CodeMagics"
```

Рисунок 6 – Использование команды %lsmagic

Команда `%lsmagic` позволяет просмотреть список доступных магических команд.

```
[13]: %env TEST=5  
env: TEST=5
```

Рисунок 7 – Создание переменной окружения с помощью команды `%new`

```
[15]: %run ./run.ipynb  
Hello world!
```

Рисунок 8 – Запуск кода из другого ноутбука с помощью команды `%run`

```
[48]: %%time  
import time  
for i in range(50):  
    time.sleep(0.01)  
  
CPU times: total: 0 ns  
Wall time: 517 ms
```

Рисунок 9 – Получение информации о времени работы кода в рамках ячейки с помощью команды `%%time`

```
[20]: %timeit x=[(i**10) for i in range(10)]  
800 ns ± 29.7 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
```

Рисунок 10 – Получение информации о среднем значении трех наиболее быстрых прогонов с помощью команды `%timeit`

### Работа в Jupyter Lab:

Создадим новую тетрадь, для этого в меню выберем File => New => Notebook.

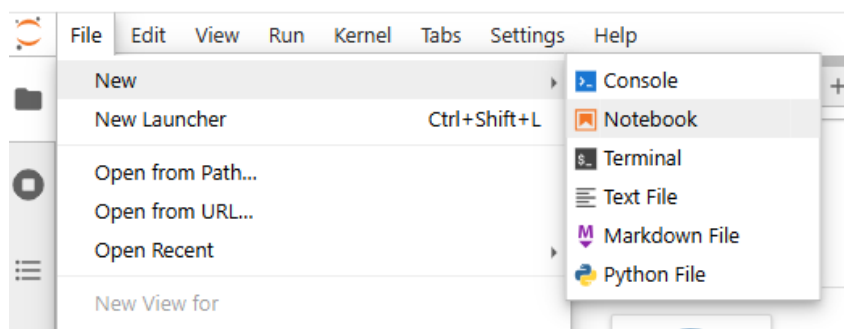
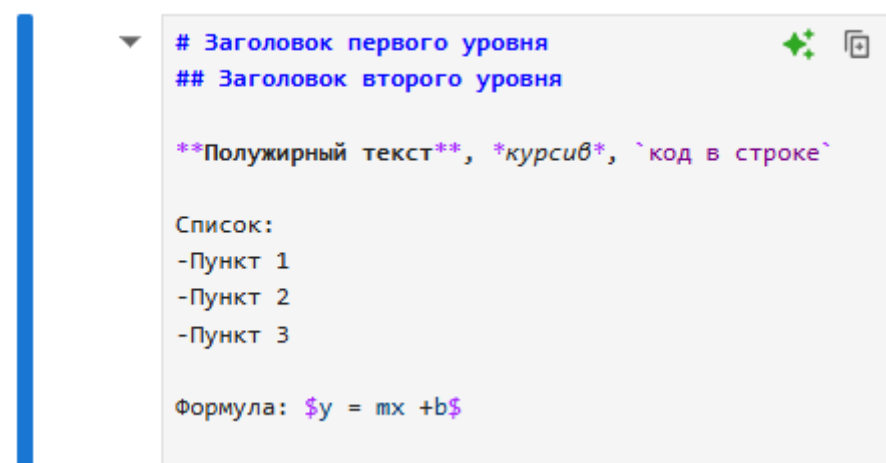


Рисунок 11 – Создание новой тетради

Для форматирования текста и формул используем Markdown.



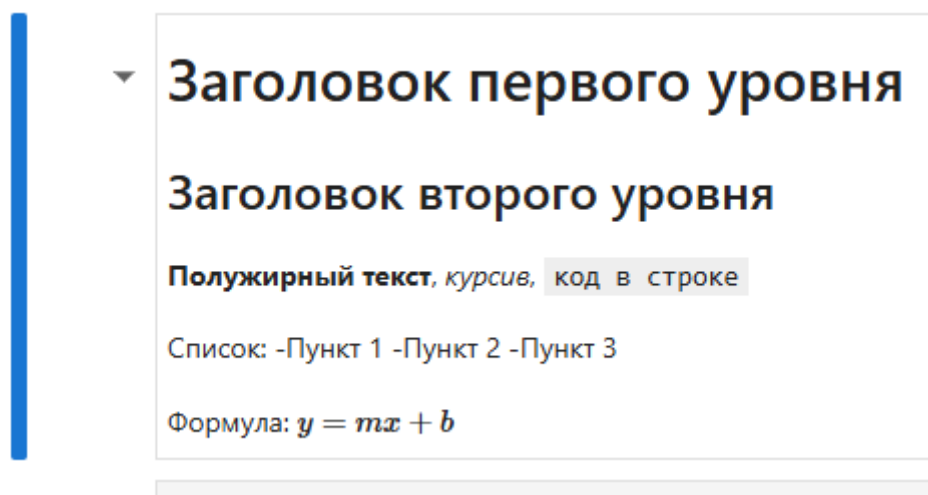
```
# Заголовок первого уровня
## Заголовок второго уровня

**Полужирный текст**, *курсив*, `код в строке`

Список:
-Пункт 1
-Пункт 2
-Пункт 3

Формула: $y = mx + b$
```

Рисунок 12 – Синтаксис для форматирования текста Markdown



```
▼ Заголовок первого уровня

Заголовок второго уровня

Полужирный текст, курсив, код в строке

Список: -Пункт 1 -Пункт 2 -Пункт 3

Формула:  $y = mx + b$ 
```

Рисунок 13 – Отформатированный текст Markdown

В отличие от Jupyter Notebook, Jupyter Lab поддерживает открытие, редактирование и анализ различных типов файлов. Загрузим таблицу CSV и посмотрим ее содержимое.



```
[25]: import pandas as pd

df = pd.read_csv("test.csv")
df.head()
```

[25]: Ryabinin;Egor

Рисунок 14 – Работа с файлами в JupyterLab

JupyterLab поддерживает динамическую визуализацию с помощью matplotlib, seaborn и plotly. На этот раз нам не придется использовать

специальные команды для отрисовки графиков, как в Jupyter Notebook, т.к по умолчанию JupyterLab отображает их в ячейке.

```
[31]: import matplotlib.pyplot as plt
import numpy as np

x=np.linspace(0,10)
y=np.sin(x)

plt.plot(x,y)
plt.xlabel("X")
plt.ylabel("Y")
plt.title("График синусоиды")
plt.show
```

```
[31]: <function matplotlib.pyplot.show(close=None, block=None)>
```

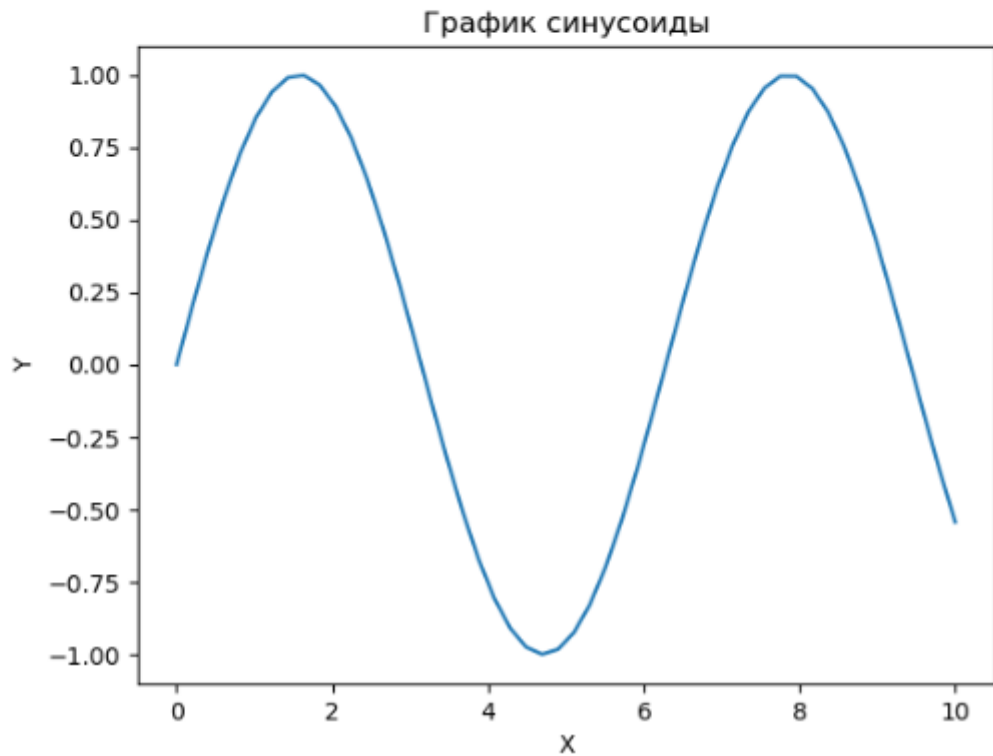


Рисунок 15 – Отображение графика в JupyterLab

Откроем терминал и выполним установку библиотек и проверку версии Python.

```
PS C:\Users\4isto> pip install numpy
Requirement already satisfied: numpy in c:\users\4isto\anaconda3\lib\site-packages (
1.26.4)
PS C:\Users\4isto> python --version
Python 3.12.7
PS C:\Users\4isto> □
```

Рисунок 16 – Пример использования команд в терминале

Используя и сравнив эти две среды, можно сделать вывод, что JupyterLab значительно превосходит Jupyter Notebook по функционалу. Однако, если требуется выполнение кода в тетрадях, Jupyter Notebook может быть более простым и удобным вариантом.

### **Работа в Google Colab:**

Google Colab позволяет бесплатно использовать GPU и TPU.

GPU может быть полезен для задач, требующих сложных математических вычислений, которые занимают много времени.

TPU применяется для тренировки нейросетей и демонстрирует высочайшую производительность даже при больших объёмах вычислительных задач.

Проверить наличие GPU и TPU можно при помощи команд:

```
[ ] import torch  
    print(torch.cuda.is_available())
```

⇒ True

Рисунок 17 – Проверка наличия GPU

```
[6] import tensorflow as tf  
     import os  
     print("TPU доступен:", "Yes" if 'COLAB_TPU_ADDR' in os.environ else "No")
```

⇒ TPU доступен: Yes

Рисунок 18 – Проверка наличия TPU

Google Colab позволяет работать с файлами на Google Диске и загружать файлы в локальное окружение.

```
[7] from google.colab import drive  
     drive.mount('/content/drive')
```

⇒ Mounted at /content/drive

Рисунок 19 – Подключение Google Drive

Файлы в Google Colab можно загружать с помощью Python.



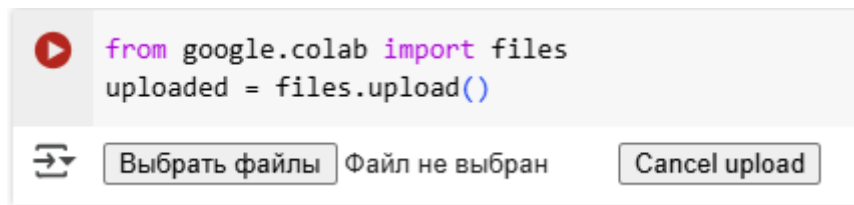


Рисунок 20 – Загрузка файлов в Google Colab с помощью Python

### Задания практической работы:

#### 1. Создайте новую **Markdown-ячейку** и:

- Напишите заголовок "Практическое задание №1".
- Добавьте **жирный** и *курсивный* текст.
- Создайте нумерованный и маркированный списки.
- Вставьте согласно индивидуального задания, например:

$$a^2 + b^2 = c^2$$

Рисунок 20 – Задание №1

#### 12. **Определение обратной матрицы (если детерминант ненулевой):**

$$A^{-1} = \frac{1}{\det(A)} \text{adj}(A).$$

Рисунок 21 – Индивидуальное задание

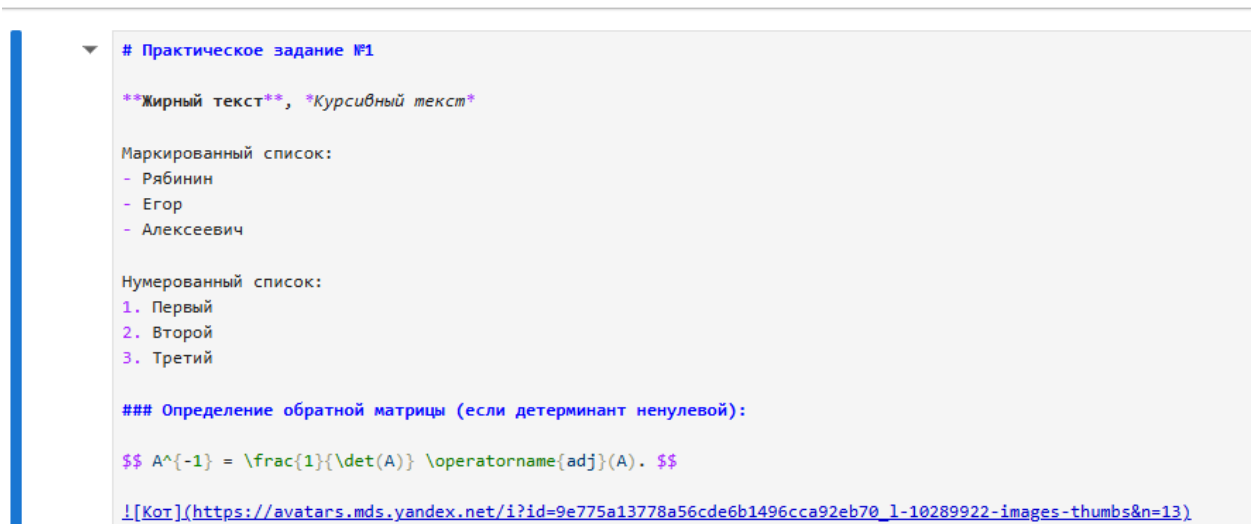


Рисунок 22 – Markdown-ячейка

## Практическое задание №1

**Жирный текст**, *Курсивный текст*

Маркированный список:

- Рябинин
- Егор
- Алексеевич

Нумерованный список:

1. Первый
2. Второй
3. Третий

Определение обратной матрицы (если детерминант ненулевой):

$$A^{-1} = \frac{1}{\det(A)} \text{adj}(A).$$



Рисунок 23 – Отформатированная Markdown-ячейка

## 2. Создайте ячейку Python-кода и:

- Запросите у пользователя его имя с помощью `input()`.
- Выведите приветствие: "Привет, <имя>! Добро пожаловать в JupyterLab / Google Colab!".
- Запустите ячейку ( `Shift + Enter` ).

Рисунок 24 – Задание №2

```
[52]: name = input("Введите ваше имя: ")
      print(f"Привет, {name}! Добро пожаловать в JupyterLab!")

Введите ваше имя: Егор
Привет, Егор! Добро пожаловать в JupyterLab!
```

Рисунок 25 – Python-код

Ссылка на репозиторий **GitHib**: [https://github.com/EgorGorilla/AI\\_and\\_ML](https://github.com/EgorGorilla/AI_and_ML)

## Контрольные вопросы:

### 1. Какие основные отличия JupyterLab от Jupyter Notebook?

JupyterLab — это более мощная и модульная версия Jupyter Notebook с поддержкой вкладок, окон, текстового редактора, терминала и других

инструментов. В Jupyter Notebook интерфейс более простой, с одной колонкой, содержащей ноутбук.

## **2. Как создать новую рабочую среду (ноутбук) в JupyterLab?**

В JupyterLab можно создать новый ноутбук через «File» → «New» → «Notebook» или нажать на значок «+» и выбрать «Notebook» в Launcher.

## **3. Какие типы ячеек поддерживаются в JupyterLab и как их переключать?**

Поддерживаются ячейки кода, Markdown и Raw. Переключение — через меню «Cell» → «Cell Type» или горячие клавиши (например, Esc + M для Markdown, Esc + Y для кода).

## **4. Как выполнить код в ячейке и какие горячие клавиши для этого используются?**

Код выполняется нажатием Shift + Enter или кнопки «Run» в панели инструментов. Ctrl + Enter выполняет без перехода к следующей ячейке, Alt + Enter выполняет и вставляет новую ячейку.

## **5. Как запустить терминал или текстовый редактор внутри JupyterLab?**

Терминал и текстовый редактор запускаются через Launcher (значок «+») или «File» → «New» → «Terminal»/«Text File».

## **6. Какие инструменты JupyterLab позволяют работать с файлами и структурами каталогов?**

В JupyterLab есть файловый браузер в левой панели, который позволяет управлять файлами и каталогами. Можно загружать, удалять, переименовывать файлы.

## **7. Как можно управлять ядрами (kernels) в JupyterLab?**

Ядра управляются через «Kernel» → «Restart Kernel», «Interrupt Kernel», «Shut Down Kernel» и через панель «Running Terminals and Kernels».

## **8. Каковы основные возможности системы вкладок и окон в интерфейсе JupyterLab?**

JupyterLab поддерживает систему вкладок и окон, позволяя работать с несколькими ноутбуками, терминалами и текстовыми файлами одновременно, перетаскивать и организовывать их.

**9. Какие магические команды можно использовать в JupyterLab для измерения времени выполнения кода? Приведите примеры.**

`%time` измеряет время выполнения одной строки, `%%time` измеряет время выполнения всей ячейки, `%timeit` и `%%timeit` выполняют код несколько раз и показывают среднее время выполнения.

**10. Какие магические команды позволяют запускать код на других языках программирования в JupyterLab?**

`%magic %lsmagic` показывает список доступных магических команд. `%script` позволяет запускать код на других языках, например, `%%bash`, `%%perl`, `%%ruby`, `%%python3`.

**11. Какие основные отличия Google Colab от JupyterLab?**

Google Colab — облачный сервис, не требует локальной установки. Поддерживает GPU, TPU, интеграцию с Google Drive. JupyterLab работает локально, требует установки и настройки.

**12. Как создать новый ноутбук в Google Colab?**

В Google Colab новый ноутбук создается через «Файл» → «Создать новый блокнот»

**13. Какие типы ячеек доступны в Google Colab, и как их переключать?**

Доступны ячейки кода и текстовые (Markdown). Переключение через меню или с помощью `Ctrl + M + M` (Markdown), `Ctrl + M + Y` (код).

**14. Как выполнить код в ячейке Google Colab и какие горячие клавиши для этого используются?**

Код выполняется `Shift + Enter`, `Ctrl + Enter` выполняет без перехода, `Alt + Enter` выполняет и добавляет новую ячейку.

**15. Какие способы загрузки и сохранения файлов поддерживает Google Colab?**

Файлы можно загружать с компьютера, работать с Google Drive, скачивать файлы командой `!wget`, сохранять результаты в Google Drive.

#### **16. Как можно подключить Google Drive к Google Colab и работать с файлами?**

Подключить Google Drive можно через `from google.colab import drive; drive.mount('/content/drive')`.

#### **17. Какие команды используются для загрузки файлов в Google Colab из локального компьютера?**

Файлы загружаются командой `from google.colab import files; files.upload()`.

#### **18. Как посмотреть список файлов, хранящихся в среде Google Colab?**

Список файлов можно посмотреть командой `!ls` или в панели файлового менеджера в левой части интерфейса.

#### **19. Какие магические команды можно использовать в Google Colab для измерения времени выполнения кода? Приведите примеры.**

`%time` измеряет время выполнения одной строки, `%%time` измеряет время выполнения всей ячейки, `%timeit` и `%%timeit` выполняют код несколько раз и показывают среднее время выполнения.

#### **20. Как можно изменить аппаратные ресурсы в Google Colab (например, переключиться на GPU)?**

Изменить аппаратные ресурсы можно через «Среда выполнения» → «Сменить среду выполнения» и выбрать GPU или TPU.

**Вывод:** в ходе практической работы мы исследовали базовые возможности интерактивных оболочек Jupyter Notebook, JupyterLab и Google Colab для языка программирования Python.